NSC Functionality in Open Stack

# Purpose

The purpose of this document is to provide complete understanding of NSC configuration and functionality in Open Stack environment. This document also describes why we need a NSC plugin, in place of openstack’s neutron openvswitch plugin. And also it points out to the changes in procedure for installation and configuration as described in different documents written.

# Prerequisite and Assumption.

This document assumes that the reader is familiar with openstack concept, architecture and installation method. And also familiarization with in house open security controller and J-stack functionality is required.

* Openstack architecture
* J-stack installation(if using j-stack)
* Open virtual Switch functionality and configuration
* Neutron Openvswitch L2 agent

# Why NSC plugin.

Openstack network node architecture use modular layer (ML2) agent for l2 data link layer functionality. So the compute node where all the VM instances resides, have l2 plugin called neutron-openvswitch-plugin to configure all the network architecture on the compute node for the VMs. This plugin creates default ovs bridges and monitors the ports, routers (dvr) creation, deletion and modification on the node and configures the open flow virtual switch for the ports connectivity and flow setup.

We need NSC plugin in place of neutron-openvswitch-plugin to implement Intel specific inspection ports configuration to establish proprietary data flow for protected ports on a VM. NSC is a replica of neutron-openvswitch-plugin API, with additional functionality to nail a data flow path with specific to Intel inspection port monitoring requirements. NSC uses neutron openvswitch l2 agent APIs to configure OVS.

* NSC is Replica of neutron-openvswitch-plugin.
* Additional functionality to add flows with proprietary data.
* Needs to be on par with neutron-openvswitch-plugin code.
* NSC is an extension to the neutron openvswitch l2 agent used to insert flows that redirect traffic to a SVA using VLAN or MPLS tagging.

# NSC Build and Deployment

The backport extension APIs for openvswitch are packaged from openstack liberty. OSC build process has provided Makefiles to make this extension packages for both Redhat-Kilo and Ubuntu-Juno. To make and install this packages.

Under SVN trunk work tree [https://svn-ip/ trunk/dpa/nsc/dev](https://svn-ip/%20trunk/dpa/nsc/dev) do a make and this will create packages for both Kilo and Juno for Redhat and Ubuntu respectively. Under this package directories are installed-\*.sh files to install these packages on respective operating systems.

\*For J-stack installation done through the J-stack.sh script this is process is automated as part of the script.

Once the packages are installed. The NSC plugin extension python APIs are tarball to a file.

Location : “https://svn-ip//trunk/tools/nsc-agent”

Files : nsc-agent-20150909T211454.tgz nsc-server-20150909T211454.tgz

On compute node(s), copy nsc-agent-xxxxxxxxx.tgz and run following:

**tar vzxf nsc-agent-\*.tgz -C /**

**initctl stop neutron-plugin-openvswitch-agent**

**initctl stop neutron-plugin-nsc-agent ; initctl start neutron-plugin-nsc-agent**

This will install all the NSC agent specific files. And as mentioned above by the commands,

neutron-plugin-openvswitch-agent is stopped and neutron-plugin-nsc-agent will take over the l2 agent extension functionality.

neutron-plugin-nsc-agent is a big python program that will be running as a background process with help of the script neutron-nsc-agent(it is part of the tar file).

The NSC agent code is all tarball, no files are checked into the SVN at this point of time.

There is no need to run any nsc-server agent, nsc-agent-20150909T211454.tgz should be ignored.

# NSC Agent process functionality.

Nsc agent (neutron-plugin-nsc-agent) which is extension of neutron-plugin-openvswitch-agent has same default initialization configuration. A virtual network is created in the compute node by default. This virtual network is used to connect all the VM instance ports to communicate with in the tenant network or outside of it i.e over the internet. By default there will be no VM instances running in compute node, hence virtual network by default looks like in Fig 1.

As the below figure depicts, there are bridges created in the open virtual switch by default they are br-int and br-tun. There are two patch interfaces that are used for communication between these two bridges patch-tun and patch-int are created. And establish a GRE tunneling interface towards network node.

## Nsc Packet encapsulation.

Intel implementation of inspection of port is achieved by vlan encapsulation in Ethernet packet to be inspected in both ingress and egress direction. Vlan id 1 or 2 is encapsulated in ingress and egress direction respectively. And this packets are send through GRE tunnel interface to the br-tun bridge which out of scope of this document. This flow works same with even already vlan tagged packets.

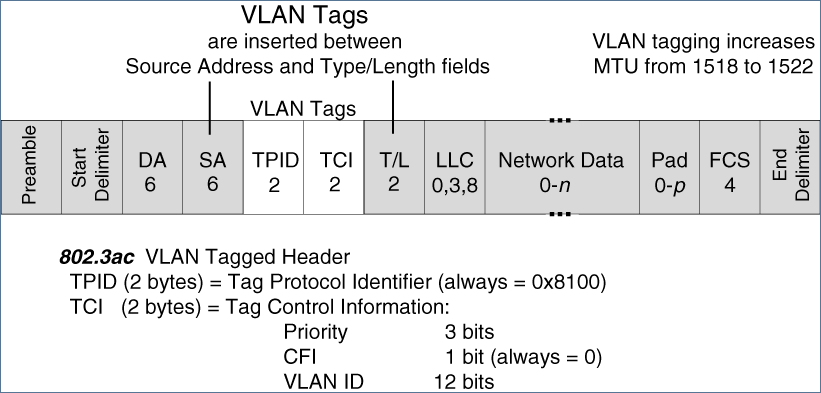
[](http://www.google.com/url?sa=i&rct=j&q=&esrc=s&source=images&cd=&cad=rja&uact=8&ved=0ahUKEwjsmKeu8qbNAhVB8WMKHWHsBH8QjRwIBw&url=http://www.wildpackets.com/resources/compendium/ethernet/ethernet_packets&psig=AFQjCNGBiiGIMDy0GMfmG-F9cpRQMlkygQ&ust=1465972243530808)

Fig 1: Default virtual network in compute node

GRE-port

Patch-int

Patch-tun

br-tun

br-int

GRE-tunneling Network node

## When VM instance is added.

NSC agent after the default initialization, will go into a loop monitoring for any port additions to begin with and thereby for any port updates. Prior to this, NSC agent will register with ovsswitchdb monitoring updates on the ovs switch ports. When a VM instance is added port will be added to ovs switch for that instance. This port addition will be detected by Nsc agent and will add the port to the br-int bridge and will create flow in ovs switch tables.Fig.2 show compute node VM instance connected to the virtual network in that node.

FIG2: VM instance connected to Virtual network in Compute node.

tap-device

VM-inst

Vm-port

Patch-tun

br-int

Patch-int

br-tun

GRE-port

GRE-tunneling towards Network node

Vm-port: 4; mac= fa: 16:3e: ae: ac: 75; tun-id=1 (each subnet have different number)

Patch-tun: 6

These rules are applied for all the packets that are received in br-int bridge. Packets that are in and out of br-tun is out of scope of this document.

INGRESS-RULE: In Yellow, read from top to bottom, top line is the first rule

That all INGRESS rules go from table120->table130

EGRESS-RULE: In Blue, read from bottom to top, bottom line is the first rule

That all EGRESS rules go from table0->table10-table130

cookie=0x1, duration=27567.7s, table=0, n\_packets=26, n\_bytes=2417, idle\_age=16108, priority=0 actions=resubmit(,10)

cookie=0x3, duration=27553.027s, table=10, n\_packets=0, n\_bytes=0, idle\_age=27553,priority=2,in\_port=4,dl\_src=fa:16:3e:ae:ac:75 actions=load:0x1->NXM\_NX\_TUN\_ID[],resubmit(,130)

cookie=0x2, duration=27558.899s, table=130, n\_packets=11, n\_bytes=1110, idle\_age=16108, priority=1,tun\_id=0x1 actions=mod\_vlan\_vid:1,output:6

cookie=0x3, duration=27555.809s, table=130, n\_packets=13, n\_bytes=1223, idle\_age=16108, priority=2,tun\_id=0x1,dl\_dst=fa:16:3e:3a:0c:0a actions=output:2

cookie=0x1, duration=931.989s, table=120, n\_packets=5513, n\_bytes=5370968, idle\_age=19, priority=0 actions=resubmit(,130)

When Service Function instance is added.

In the OSC terminology when a service appliance is deployed and IPS instance is created in the compute in the compute node. As a part of service deployment in OSC a we create three subnets i.e. demo, inspection and external networks. So a port is created on each subnet.

We are interested here in only demo port which is management port and inspection port that are used in this discussion.

As mentioned above when IPS instance is added to compute node a management and inspection port is created will be connected to virtual network on compute node. Nsc agent will detect creating of these ports and will configure the ports on the br-int bridge of the OVS i.e. virtual network and will establish a flow on the data path Fig 3 depicts the above mentioned network connectivity.

FIG3: and IPS and VM instance connected to a virtual network (through OVS) in compute node.

Eth1

Eth0

IPS-inst

Eth0

VM-inst

br-int

Vm-port

demo-port

Insp-port

Patch-tun

Patch-int

br-tun

GRE-port

GRE-tunneling towards Network node

Vm-port: 4; mac= fa: 16:3e: ae: ac: 75; tun-id=1 (each subnet have different number)

Demo-port: 8; mac = fa: 16:3e:0a:e2: Fe; tun-id=1(just to show the flow no use for the scope of this document)

Patch-tun: 6

These rules are applied for all the packets that are received in br-int bridge. Packets that are in and out of br-tun is out of scope of this document.

INGRESS-RULE: In Yellow, read from top to bottom, top line is the first rule

That all INGRESS rules go from table120->table130

EGRESS-RULE: In Blue, read from bottom to top, bottom line is the first rule

That all EGRESS rules go from table0->table10-table130

No rule is added to insp-port, why because this is not consider as data end to end port.

cookie=0x1, duration=27567.7s, table=0, n\_packets=26, n\_bytes=2417, idle\_age=16108, priority=0 actions=resubmit(,10)

cookie=0x3, duration=27553.027s, table=10, n\_packets=0, n\_bytes=0, idle\_age=27553,priority=2,in\_port=4,dl\_src=fa:16:3e:ae:ac:75 actions=load:0x1->NXM\_NX\_TUN\_ID[],resubmit(,130)

cookie=0x3, duration=811.707s, table=10, n\_packets=1660, n\_bytes=138765, idle\_age=19, priority=2,in\_port=8,dl\_src=fa:16:3e:0a:e2:fe actions=load:0x1->NXM\_NX\_TUN\_ID[],resubmit(,130)

cookie=0x2, duration=27558.899s, table=130, n\_packets=11, n\_bytes=1110, idle\_age=16108, priority=1,tun\_id=0x1 actions=mod\_vlan\_vid:1,output:6

cookie=0x3, duration=27555.809s, table=130, n\_packets=13, n\_bytes=1223, idle\_age=16108, priority=2,tun\_id=0x1,dl\_dst=fa:16:3e:3a:0c:0a actions=output:2

cookie=0x1, duration=931.989s, table=120, n\_packets=5513, n\_bytes=5370968, idle\_age=19, priority=0 actions=resubmit(,130)

When Security port group is created and bonded.

When security port group created and bonded to service deployment. The VM instance that is part of protected group, that instance data end to end port will called inspected port and a port update event will happen in the compute node on that particular port(in our case here VM-port as shown in the figure). Intel implementation of NSC uses the profile field of the updated port message structure of the network port update message and will send the inspection port that would be used to apply the service function (in our case here insp-port as shown in the figure).From this update of the port Nsc will create following flow from the inspected port in the tables.

Vm-port: 4; mac= fa: 16:3e: ae: ac: 75; tun-id=1 (each subnet have different number)

Insp-port: 9

Patch-tun: 10

These rules are applied for all the packets that are received in br-int bridge. Packets that are in and out of br-tun is out of scope of this document.

INGRESS-RULE: In Yellow, read from top to bottom, top line is the first rule

That all INGRESS rules go from table120->table130

EGRESS-RULE: In Blue, read from bottom to top, bottom line is the first rule

That all EGRESS rules go from table0->table10-table130

After binding a rule or flow is created to the inspection port in the ovs db.

cookie=0xbade, duration=511.813s, table=0, n\_packets=0, n\_bytes=0, idle\_age=511, priority=5,ip,in\_port=4,dl\_src=fa:16:3e:ae:ac:75 actions=mod\_vlan\_vid:1,mod\_vlan\_pcp:1,output:9

cookie=0xbade, duration=511.605s, table=10, n\_packets=0, n\_bytes=0, idle\_age=511, priority=3,in\_port=9,dl\_vlan=1,dl\_vlan\_pcp=2,dl\_src=fa:16:3e:ae:ac:75,dl\_dst=01:00:00:00:00:00/01:00:00:00:00:00 actions=mod\_vlan\_vid:0,mod\_vlan\_pcp:0,strip\_vlan,load:0x1->NXM\_NX\_TUN\_ID[],load:0x4->NXM\_OF\_IN\_PORT[],resubmit(4,130)

cookie=0x2, duration=1283.804s, table=130, n\_packets=474, n\_bytes=57722, idle\_age=3, priority=1,tun\_id=0x1 actions=mod\_vlan\_vid:1,output:10

cookie=0xbade, duration=511.708s, table=120, n\_packets=0, n\_bytes=0, idle\_age=511, priority=5,ip,dl\_dst=fa:16:3e:ae:ac:75 actions=mod\_vlan\_vid:1,mod\_vlan\_pcp:2,output:9

cookie=0xbade, duration=511.657s, table=10, n\_packets=0, n\_bytes=0, idle\_age=511, priority=2,in\_port=9,dl\_vlan=1,dl\_vlan\_pcp=2,dl\_dst=fa:16:3e:ae:ac:75 actions=mod\_vlan\_vid:0,mod\_vlan\_pcp:0,strip\_vlan,load:0x1->NXM\_NX\_TUN\_ID[],resubmit(,130)

cookie=0x3, duration=511.306s, table=130, n\_packets=0, n\_bytes=0, idle\_age=511, priority=2,tun\_id=0x1,dl\_dst=fa:16:3e:ae:ac:75 actions=output:4

## Nsc Statistics

The NSC agent also collects and stores statistics for each bridge port. This statistics are reported periodically.

## NSC agent port configuration database

All the ports that are created and updated will be stored in the port-info data structure of the agent process.